kubernetes1.13安装

环境准备

----------环境信息----------
192.168.1.111 master
192.168.1.112 node1
192.168.1.113 node2

配置好主机名/etc/hosts（所有节点）

```
cat > /etc/test <<EOF
127.0.0.1       localhost
192.168.1.111 master
192.168.1.112 node1
192.168.1.113 node2
EOF
```

关闭SELinux和防火墙（所有节点）

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
setenforce 0
systemctl stop firewalld && systemctl disable firewalld
```

关闭swap（所有节点）

```
swapoff -a
#限制开机启动
sed -i 's/\/dev\/mapper\/centos-swap/#\/dev\/mapper\/centos-swap/g' /etc/fstab
```

配置各节点系统内核参数使流

过网桥的流量也进入iptables/netfilter框架中（所有节点）

```
#解决sysctl -p生效时报错 执行以下三句命令
yum install -y bridge-utils.x86_64
modprobe bridge
modprobe br_netfilter
#以上执行完再执行下面的指令  以免报错
cat <<EOF >  /etc/sysctl.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl -p
```

配置阿里K8S YUM源（所有节点）

```
cat > /etc/yum.repos.d/kubernetes.repo <<EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
EOF
```

组件安装

docker安装（所有节点）

```
yum -y install docker
systemctl enable docker && systemctl start docker
```

配置阿里镜像加速器 如果有私仓需要信任配置再启用第二句

```
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://h9dkpwpe.mirror.aliyuncs.com"],
   # "insecure-registries":["192.168.53.114"]
}
EOF
systemctl daemon-reload
systemctl restart docker
```

kubeadm安装（所有节点）

默认安装最新版本
#yum install -y kubelet kubeadm kubectl
此处指定版本13.3：
yum install -y kubelet-1.13.3-0 kubeadm-1.13.3-0 kubectl-1.13.3-0

kubelet设置开机启动 并添加配置解决报错问题（所有节点）

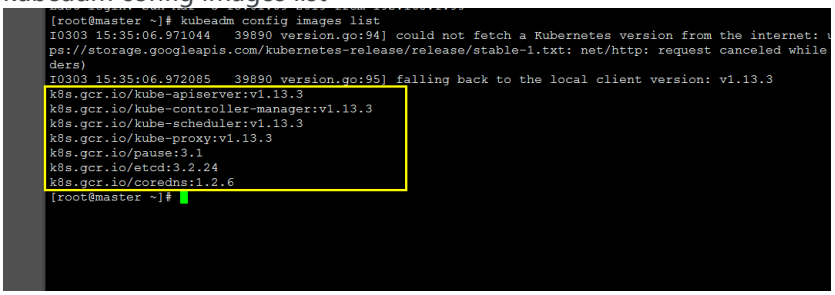
systemctl enable kubelet
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
添加以下内容：
Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=systemd --runtime-cgroups=/systemd/system.slice --kubelet-cgroups=/systemd/system.slice"

systemctl start kubelet
systemctl enable kubelet

下载镜像
主节点镜像列表 （master)

#查看所需的镜像及版本 附上可视化的镜像dashboard

#flannel镜像可以直接下，这里不作处理 这里主要解决google镜像无法下载问题
kubeadm config images list



docker pull mirrorgooglecontainers/kube-apiserver-amd64:v1.13.3
docker pull mirrorgooglecontainers/kube-controller-manager-amd64:v1.13.3
docker pull mirrorgooglecontainers/kube-scheduler-amd64:v1.13.3
docker pull mirrorgooglecontainers/kube-proxy-amd64:v1.13.3
docker pull mirrorgooglecontainers/pause-amd64:3.1
docker pull mirrorgooglecontainers/etcd-amd64:3.2.24
docker pull mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1
docker pull coredns/coredns:1.2.6
或者： carlziess/coredns-1.2.6

docker tag mirrorgooglecontainers/kube-apiserver-amd64:v1.13.3 k8s.gcr.io/kube-apiserver:v1.13.3
docker tag mirrorgooglecontainers/kube-controller-manager-amd64:v1.13.3 k8s.gcr.io/kube-controller-manager:v1.13.3
docker tag mirrorgooglecontainers/kube-scheduler-amd64:v1.13.3 k8s.gcr.io/kube-scheduler:v1.13.3
docker tag mirrorgooglecontainers/kube-proxy-amd64:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause-amd64:3.1 k8s.gcr.io/pause:3.1
docker tag mirrorgooglecontainers/etcd-amd64:3.2.24 k8s.gcr.io/etcd:3.2.24
docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6
docker tag mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1

运行节点镜像列表 (node1、node2)

#flannel镜像可以直接下，这里不作处理 这里主要解决google镜像无法下载问题
docker pull mirrorgooglecontainers/kube-proxy-amd64:v1.13.3
docker pull mirrorgooglecontainers/pause-amd64:3.1

docker tag mirrorgooglecontainers/kube-proxy-amd64:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause-amd64:3.1 k8s.gcr.io/pause:3.1

主节点初始化

kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=0.0.0.0

```
1
2 [root@master ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-ad
dress=0.0.0.0
3 [init] Using Kubernetes version: v1.13.3
4 [preflight] Running pre-flight checks
5 error execution phase preflight: [preflight] Some fatal errors occurred:
6  [ERROR Swap]: running with swap on is not supported. Please disable swap
7 [preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
8 [root@master ~]# swapoff -a
9 [root@master ~]# #限制开机启动
10 [root@master ~]# sed -i 's/\/dev\/mapper\/centos-swap/#\/dev\/mapper\/centos-swap/g' /etc/fstab
11 [root@master ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-a
ddress=0.0.0.0
12 [init] Using Kubernetes version: v1.13.3
13 [preflight] Running pre-flight checks
14 [preflight] Pulling images required for setting up a Kubernetes cluster
15 [preflight] This might take a minute or two, depending on the speed of your internet connection
16 [preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
17 [kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
18 [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
19 [kubelet-start] Activating the kubelet service
20 [certs] Using certificateDir folder "/etc/kubernetes/pki"
21 [certs] Generating "etcd/ca" certificate and key
22 [certs] Generating "etcd/peer" certificate and key
23 [certs] etcd/peer serving cert is signed for DNS names [master localhost] and IPs [192.168.1.111 127.0.0.1 ::1]
24 [certs] Generating "etcd/healthcheck-client" certificate and key
25 [certs] Generating "etcd/server" certificate and key
26 [certs] etcd/server serving cert is signed for DNS names [master localhost] and IPs [192.168.1.111 127.0.0.1 ::1]
27 [certs] Generating "apiserver-etcd-client" certificate and key
28 [certs] Generating "ca" certificate and key
29 [certs] Generating "apiserver-kubelet-client" certificate and key
30 [certs] Generating "apiserver" certificate and key
31 [certs] apiserver serving cert is signed for DNS names [master kubernetes kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.111]
32 [certs] Generating "front-proxy-ca" certificate and key
```

```
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 22.004987 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the configuration for the kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "master" as an annotation
[mark-control-plane] Marking the node master as control-plane by adding the label "node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node master as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: j9fg3l.0fxvkxmxgtjyrgxn
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.1.111:6443 --token j9fg3l.0fxvkxmxgtjyrgxn --discovery-token-ca-cert-hash sha256:f2809ca9baf77d868444112a3ec1b28a0bcf5bc18e55f1fa5a6aefe72c170e8d

[root@master ~]#
```

执行 提示内容：

MASTER节点

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

NODE节点

kubeadm join 192.168.1.111:6443 --token j9fg3l.0fxvkxmxgtjyrgxn --discovery-token-ca-cert-hash sha256:f2809ca9baf77d868444112a3ec1b28a0bcf5bc18e55f1fa5a6aefe72c170e8d

查看节点信息 目前NotReady是因为fl网络未配的原因，这里显示这样就对了

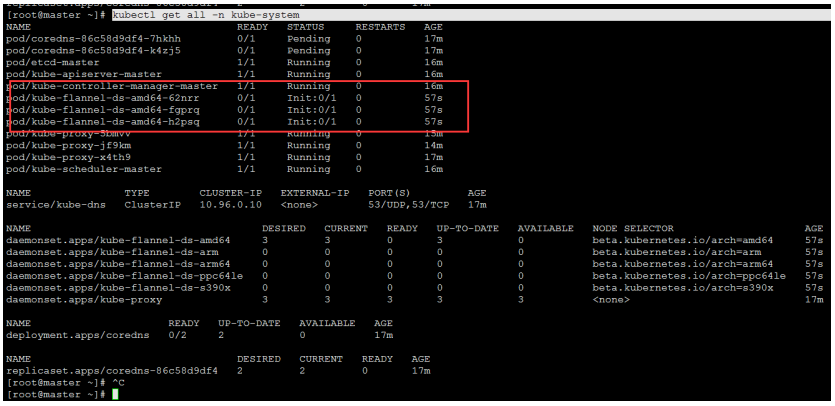
[root@master ~]# kubectl get node
NAME STATUS ROLES AGE VERSION
master NotReady master 3m v1.13.3
node1 NotReady <none> 47s v1.13.3
node2 NotReady <none> 8s v1.13.3
[root@master ~]#

安装flannel网络（主节点）

查看网络组件的版本：https://github.com/coreos/flannel/blob/master/Documentation/kube-flannel.yml 最新版为11注意最后一句话修改版本号就可以apply上

```
mkdir -p /etc/cni/net.d/

cat <<EOF> /etc/cni/net.d/10-flannel.conf
{
“name”: “cbr0”,
“type”: “flannel”,
“delegate”: {
“isDefaultGateway”: true
}
}

EOF

mkdir /usr/share/oci-umount/oci-umount.d -p

mkdir /run/flannel/

cat <<EOF> /run/flannel/subnet.env
FLANNEL_NETWORK=10.244.0.0/16
FLANNEL_SUBNET=10.244.1.0/24
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true

EOF

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.11.0/Documentation/kube-flannel.yml

```

这个过程会比较慢，因为需要去现pull flannel的 docker镜像
kubectl get all -n kube-system
可以看到正在初始化

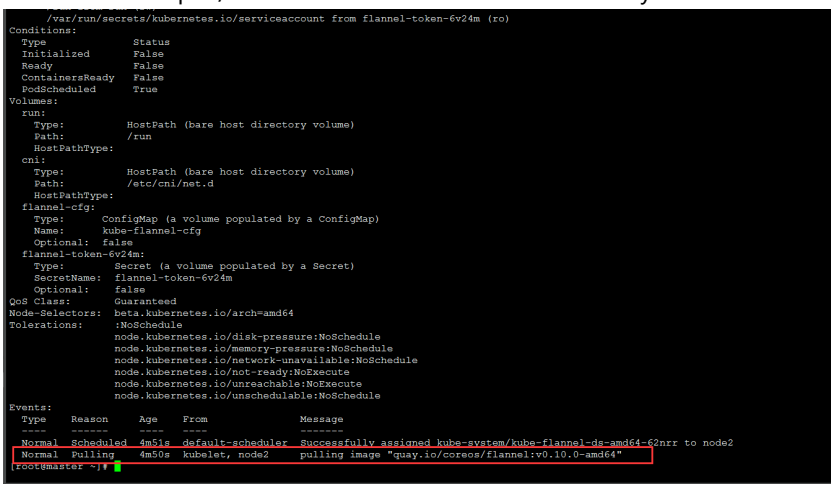


一段时间后 只有一个节点Ready

```
[root@master ~]# kubectl get node
NAME STATUS ROLES AGE VERSION
master NotReady master 19m v1.13.3
node1 Ready <none> 17m v1.13.3
node2 NotReady <none> 16m v1.13.3
```

查看正在初始化的pod信息，可以看到还在pull镜像

kubectl describe pod/kube-flannel-ds-amd64-62nrr -n kube-system



过了三五分钟后，全部Ready状态了

```
[root@master ~]# kubectl get node
NAME STATUS ROLES AGE VERSION
master Ready master 22m v1.13.3
node1 Ready <none> 20m v1.13.3
node2 Ready <none> 19m v1.13.3
[root@master ~]#

```

```
#健康检查
[root@master ~]# kubectl get cs
NAME STATUS MESSAGE ERROR
scheduler Healthy ok
controller-manager Healthy ok
etcd-0 Healthy {"health": "true"}
[root@master ~]#

```

重启所有节点，看开机是否正常运行

```
[root@master ~]# kubectl get cs
NAME STATUS MESSAGE ERROR
```

```
scheduler Healthy ok
controller-manager Healthy ok
etcd-0 Healthy {"health": "true"}
[root@master ~]#
```

接下我们安装可视化界面

Kubernetes-DashBoard用户界面

第一步：安装
第二步：建账号
第三步：获取管理密码
第四步：WEB登录管理

第一步：安装

wget https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml

修改内容：将以下国内不可达的镜像地址改为docker hub可达的地址

```
vi kubernetes-dashboard.yaml
```

/****
#文件尾部添加红色 开启node端的访问入口：
#这里的坑就是大小写和间距 不能有TAB全部用空格对好！、
修改第一处：
image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
改为
image: mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1
修改第二处：
type: NodePort
ports:
- port: 443
targetPort: 8443
nodePort: 30001
****/

```
[root@master ~]# kubectl apply -f kubernetes-dashboard.yaml
secret/kubernetes-dashboard-certs created
serviceaccount/kubernetes-dashboard created
role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
deployment.apps/kubernetes-dashboard created
service/kubernetes-dashboard created
```

修改完文件内容如下附件

kubernetes-dashbo..d.yaml
4.51KB

第二步：创建管理员账号

创建授权账号登录：admin-user.yaml

```
[root@master2 k8s]# vi admin-k8s.yaml
apiVersion: v1
kind: ServiceAccount
```

```
metadata:
 name: admin-user
 namespace: kube-system

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: admin-user
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- kind: ServiceAccount
 name: admin-user
 namespace: kube-system
```

```
[root@master ~]# kubectl apply -f admin-k8s.yaml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
[root@master ~]#
```

第三步获取token密码：

指令：

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

```
[root@master ~]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
Name: admin-user-token-vwkbc
Namespace: kube-system
Labels: <none>
Annotations: kubernetes.io/service-account.name: admin-user
 kubernetes.io/service-account.uid: 877d81bf-3d8f-11e9-bb35-000c298e17b8

Type: kubernetes.io/service-account-token

Data
====
ca.crt: 1025 bytes
namespace: 11 bytes
token: eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXZ3a2JjIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI4NzdkODFiZi0zZDhmLTExZTktYmIzNS0wMDBjMjk4ZTE3YjgiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.rQo5mLr1Akrx-gLpgKySpR4h1ieZE2w4SGmucbCqP1vK2osR6R434_dNgp3ML1_v02FBysTZbPoKa8FnIxOkuN2wzNfhmwZBuwCfLI2oJzUUvUbmvGmfmjdrVEsQqpMvQ9b392nE7CXYuwLYnzNVNxJWETn5MH2JXyVLrzsW8eSO_RMcWQA0xGfqSHuCc7aFgwcSpbd50nmb2wBXg0Ji2fz9CRvtDzakpMHop-vXR6JngopxleYxTfALbjL-gGyUr0O501uoXfZhYGJzcWxELYI9PWcbewSZkr6oyriLlkRzZVy0PZe0e179ThhnuO6rmWy9-eVM1o2LOBusPS8jBw
[root@master ~]#
```

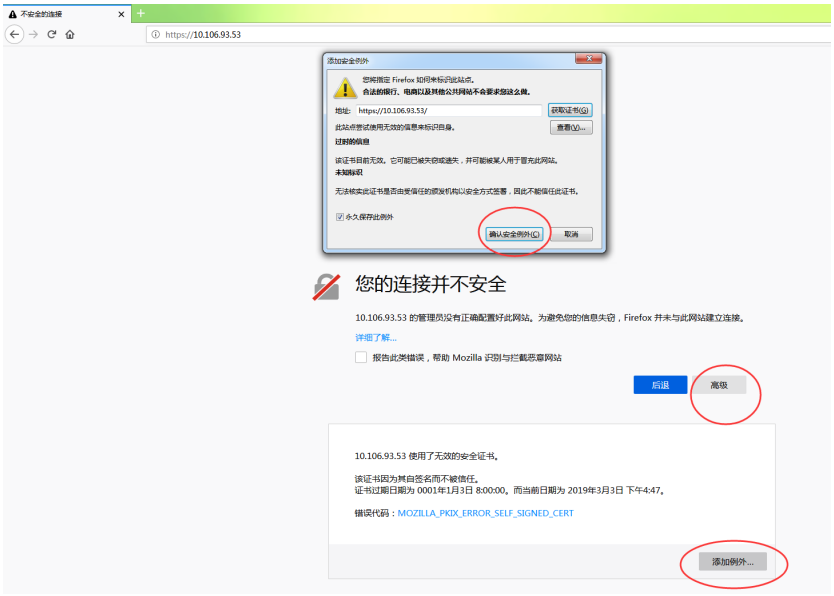
第四步密码登录：

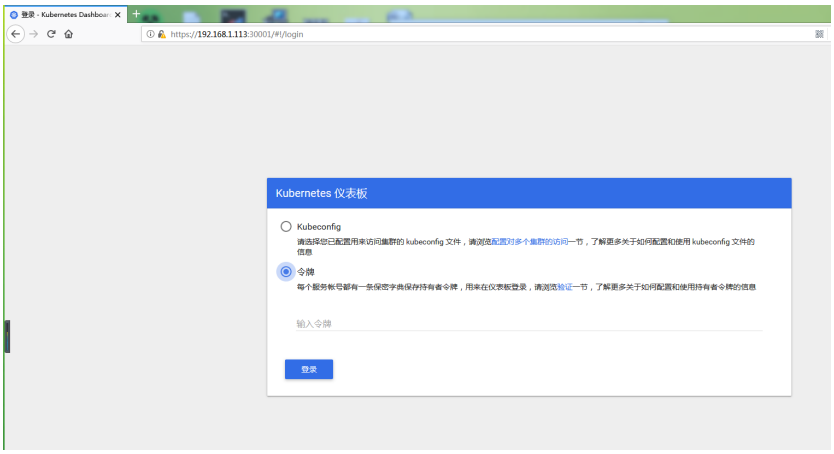
```
[root@master ~]# kubectl get svc -n kube-system -o wide
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
kube-dns ClusterIP 10.96.0.10 <none> 53/UDP,53/TCP 53m k8s-app=kube-dns
kubernetes-dashboard NodePort 10.106.93.53 <none> 443:30001/TCP 10m k8s-app=kubernetes-dashboard

[root@master ~]# kubectl get pods -n kube-system -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
kubernetes-dashboard-76479d66bb-hk2pb 1/1 Running 0 11m 10.244.2.2 node2 <none> <none>

```

查得IP为：10.106.93.53 运行于node2
添加路由访问：https://10.106.93.53
或者用NodePort访问：https://192.168.1.113:30001
注意点：此处最好用火狐 添加不安全网站例外后可以正常访问 密码就是上面的token

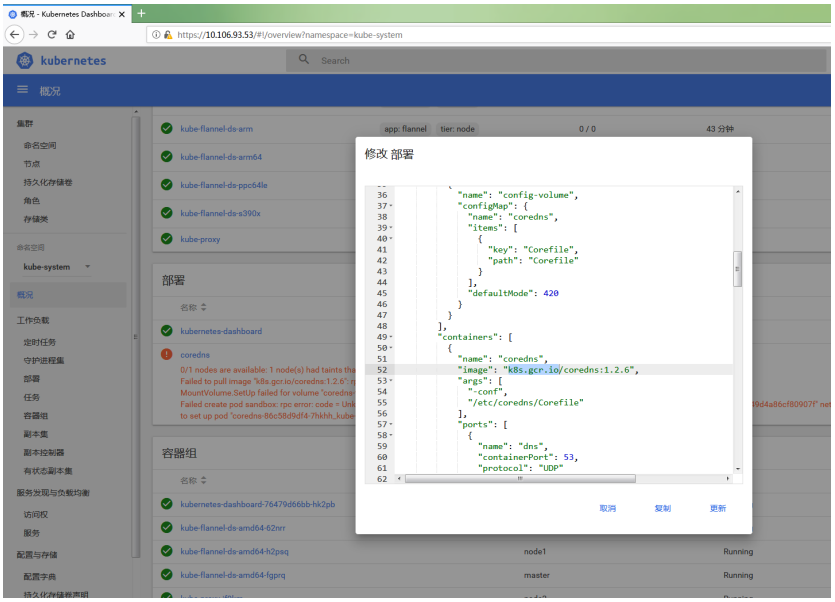
添加路由

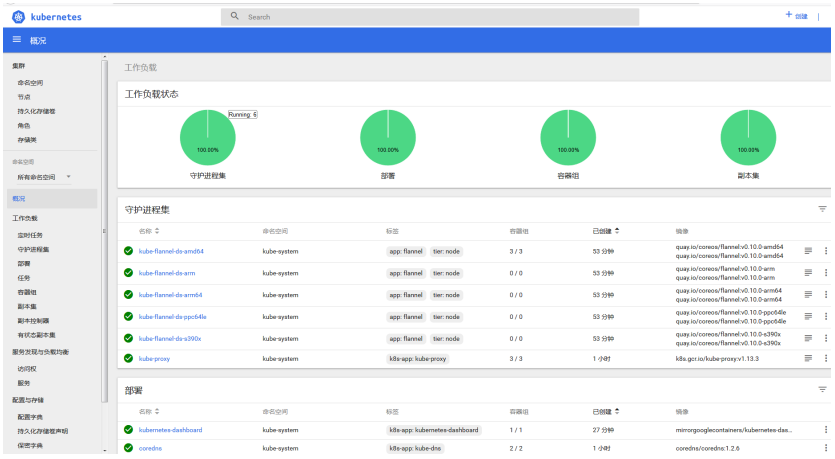
```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>route add 10.106.0.0 mask 255.255.0.0 192.168.1.111
  操作完成!
```





登录后发现还有容器因为翻墙问题出错，改掉它为： coredns/coredns:1.2.6 更新后一切完美





解决DNS问题

干掉core-dns
kubectl delete deployment.apps/coredns -n kube-system
kubectl delete svc coredns -n kube-system
kubectl delete svc kube-dns -n kube-system
官网文件：https://github.com/kubernetes/kubernetes/blob/release-1.8/cluster/addons/dns/kubedns-controller.yaml.base

wget https://github.com/kubernetes/kubernetes/blob/release-1.8/cluster/addons/dns/kubedns-controller.yaml.base
mv kubedns-controller.yaml.base kube-dns.yaml

里面修改了域名和IP地址.以下这两步已完成替换在下面的yaml内容里

```
sed -i 's/__PILLAR__DNS__SERVER__/10.96.0.10/g' kube-dns.yaml
sed -i 's/__PILLAR__DNS__DOMAIN__/cluster.local/g' kube-dns.yaml

```

```
vi kube-dns.yaml
# Copyright 2016 The Kubernetes Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
```

```
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Should keep target in cluster/addons/dns-horizontal-autoscaler/dns-horizontal-autoscaler.yaml
# in sync with this file.

# __MACHINE_GENERATED_WARNING__

apiVersion: v1
kind: Service
metadata:
 name: kube-dns
 namespace: kube-system
 labels:
 k8s-app: kube-dns
 kubernetes.io/cluster-service: "true"
 addonmanager.kubernetes.io/mode: Reconcile
 kubernetes.io/name: "KubeDNS"
spec:
 selector:
 k8s-app: kube-dns
 clusterIP: 10.96.0.10
 ports:
 - name: dns
 port: 53
 protocol: UDP
 - name: dns-tcp
 port: 53
 protocol: TCP
---
apiVersion: v1
kind: ServiceAccount
metadata:
 name: kube-dns
 namespace: kube-system
 labels:
 kubernetes.io/cluster-service: "true"
 addonmanager.kubernetes.io/mode: Reconcile
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: kube-dns
 namespace: kube-system
 labels:
 addonmanager.kubernetes.io/mode: EnsureExists
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: kube-dns
 namespace: kube-system
 labels:
 k8s-app: kube-dns
 kubernetes.io/cluster-service: "true"
 addonmanager.kubernetes.io/mode: Reconcile
```

```
spec:
  # replicas: not specified here:
  # 1. In order to make Addon Manager do not reconcile this replicas parameter.
  # 2. Default is 1.
  # 3. Will be tuned in real time if DNS horizontal auto-scaling is turned on.
  strategy:
  rollingUpdate:
  maxSurge: 10%
  maxUnavailable: 0
  selector:
  matchLabels:
  k8s-app: kube-dns
  template:
  metadata:
  labels:
  k8s-app: kube-dns
  annotations:
  scheduler.alpha.kubernetes.io/critical-pod: ''
  seccomp.security.alpha.kubernetes.io/pod: 'docker/default'
  spec:
  priorityClassName: system-cluster-critical
  securityContext:
  supplementalGroups: [ 65534 ]
  fsGroup: 65534
  tolerations:
  - key: "CriticalAddonsOnly"
  operator: "Exists"
  volumes:
  - name: kube-dns-config
  configMap:
  name: kube-dns
   optional: true
   containers:
   - name: kubedns
   image: yxmu2006/k8s-dns-kube-dns:1.14.13
   resources:
   # TODO: Set memory limits when we've profiled the container for large
   # clusters, then set request = limit to keep this container in
   # guaranteed class. Currently, this container falls into the
   # "burstable" category so the kubelet doesn't backoff from restarting it.
   limits:
   memory: 170Mi
   requests:
   cpu: 100m
   memory: 70Mi
   livenessProbe:
   httpGet:
   path: /healthcheck/kubedns
   port: 10054
   scheme: HTTP
   initialDelaySeconds: 60
   timeoutSeconds: 5
   successThreshold: 1
   failureThreshold: 5
   readinessProbe:
   httpGet:
   path: /readiness
   port: 8081
   scheme: HTTP
   # we poll on pod startup for the Kubernetes master service and
```

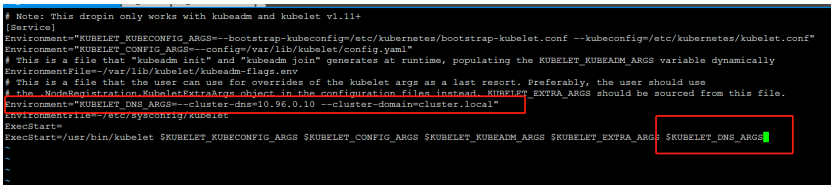
```
# only setup the /readiness HTTP server once that's available.
initialDelaySeconds: 3
timeoutSeconds: 5
args:
- --domain=cluster.local.
- --dns-port=10053
- --config-dir=/kube-dns-config
- --v=2
env:
- name: PROMETHEUS_PORT
value: "10055"
ports:
- containerPort: 10053
name: dns-local
protocol: UDP
- containerPort: 10053
name: dns-tcp-local
protocol: TCP
- containerPort: 10055
name: metrics
protocol: TCP
volumeMounts:
- name: kube-dns-config
mountPath: /kube-dns-config
- name: dnsmasq
image: yxmu2006/k8s-dns-dnsmasq-nanny:1.14.13
livenessProbe:
httpGet:
path: /healthcheck/dnsmasq
port: 10054
scheme: HTTP
initialDelaySeconds: 60
timeoutSeconds: 5
successThreshold: 1
failureThreshold: 5
args:
- -v=2
- -logtostderr
- -configDir=/etc/k8s/dns/dnsmasq-nanny
- -restartDnsmasq=true
- --
- -k
- --cache-size=1000
- --no-negcache
- --dns-loop-detect
- --log-facility=-
- --server=/cluster.local/127.0.0.1#10053
- --server=/in-addr.arpa/127.0.0.1#10053
- --server=/ip6.arpa/127.0.0.1#10053
ports:
- containerPort: 53
name: dns
protocol: UDP
- containerPort: 53
name: dns-tcp
protocol: TCP
# see: https://github.com/kubernetes/kubernetes/issues/29055 for details
resources:
requests:
cpu: 150m
```

```
  memory: 20Mi
  volumeMounts:
  - name: kube-dns-config
  mountPath: /etc/k8s/dns/dnsmasq-nanny
  - name: sidecar
  image: yxmu2006/k8s-dns-sidecar:1.14.13
  livenessProbe:
  httpGet:
  path: /metrics
  port: 10054
  scheme: HTTP
  initialDelaySeconds: 60
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 5
  args:
  - --v=2
  - --logtostderr
  - --probe=kubedns,127.0.0.1:10053,kubernetes.default.svc.cluster.local,5,SRV
  - --probe=dnsmasq,127.0.0.1:53,kubernetes.default.svc.cluster.local,5,SRV
  ports:
  - containerPort: 10054
  name: metrics
  protocol: TCP
  resources:
  requests:
  memory: 20Mi
  cpu: 10m
  dnsPolicy: Default # Don't use cluster DNS.
  serviceAccountName: kube-dns
```

kubectl create -f kube-dns.yaml

修改启动参数配置文件指定 DNS
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

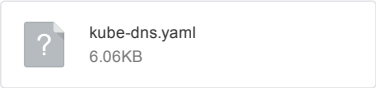
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"



busybox用于测试内容如下：

```

apiVersion: v1
kind: Service
metadata:
 name: default-subdomain
spec:
 selector:
 name: busybox
 clusterIP: None
 ports:
 - name: foo # Actually, no port is needed.
 port: 1234
```

```
 targetPort: 1234
---
apiVersion: v1
kind: Pod
metadata:
 name: busybox1
 labels:
 name: busybox
spec:
 hostname: busybox-1
 subdomain: default-subdomain
 containers:
 - image: busybox
 command:
 - sleep
 - "3600"
 name: busybox
```



kubectl create -f busybox.yaml
测试通过

```
[root@master2 ~]# kubectl exec -it busybox1 -- nslookup kubernetes
Server: 10.96.0.10
Address: 10.96.0.10:53

Non-authoritative answer:
Name: kubernetes.default.svc.cluster.local
Address: 10.96.0.1

```